head computers (three-tiered system). In a first step, the master or head computer reads a configuration file (step 520) which specifies various configuration information for the graphics display(s) in that system. From this configuration information, the various master computers configure the individual slave computers, or graphic node devices (step 530). Thereafter, the system configures the various graphic node configuration files (step 540). Thereafter, each of the graphics nodes are started (step 550) based upon their individual configuration information. Thereafter, the graphics processing is performed by the various graphics nodes (step 560). Steps 550 and 560 are conventional steps and need not be described in detail herein.

In this regard, the system and method of the present invention relates principally to the performance of steps 520, 530 and 540, and each of these steps is more particularly described in connection with the flow charts of Figs. 20, 21 and 22, respectively. Reference is now made to FIG. 20, which is a flow chart illustrating the top-level operation of the "Read Configuration File" step 520 illustrated in FIG. 19. As previously mentioned, in accordance with one embodiment of the invention, a main configuration file (e.g., head configuration file or master configuration file) contains configuration information that is used for the configuration of the various slave nodes that are configured to collectively render a single display. As a first step in the process of reading the main configuration file, a determination may be made as to whether there are nested graphics nodes (step 521). In essence, this step makes the determination as to whether the current node is a master computer (in which there are no nested graphics nodes) or a head computer (which includes nested graphics nodes). As illustrated, if the determination is made that there are, indeed, nested graphics nodes, then the method proceeds to find or identify all master graphics nodes (step 522). This step may be performed simply by scanning (by the head computer) through the configuration file to identify the specific, predetermined master nodes (which are specifically defined in the configuration file). This step also identifies any specific configuration options for the ultimate slave computers.

The method then creates configuration information for each master computer (step 523). This step essentially performs a data translation, translating information from, for example, a head configuration file into multiple master configuration files. This step builds each such master configuration file and delivers each such file to the various master computers. Alternatively, in a socket-based implementation, as described above, this step may be configured to deliver the configuration information for each master computer directly to the respective master computers through a communication port or socket. Then, the method proceeds to step 524 which recursively calls the function "initialize graphics nodes" (e.g., the flow chart of FIG. 19) for each master node identified.

If step 521 determines that the current node is a master computer, then the method proceeds to step 525, in which it finds or identifies all slave graphics nodes. This step is similar to step 522, in that the current master computer may evaluate the master configuration file to determine all associated slave nodes (which are defined in the master configuration file), as well as any specific options delineated within the master configuration file for the respective slave nodes. The method then determines, based on the information contained in the master configuration file, all "per-slave" options (step 526). In this respect, various slave computers may be configured with different options, so long as there is intercompatibility among the various slave computers to render a single display. Finally, the method identifies all global slave options (i.e., all options that are applicable to all slave computers operating under the direction of single master computer) (step 527).

After the configuration files are read and translated, and as illustrated in FIG. 19, the method proceeds to configure the various graphic node devices (step 530). This step essentially performs a data translation process from master to slave nodes in which the various slave nodes are configured to have compatible hardware configurations. This step will function as more particularly illustrated in FIG. 21. In this regard, the method creates or initializes graphics video timing information (step 532). This step essentially defines or sets hardware information such as the screen size, pixel depth, etc. The method may then install video-timing information onto the various slave nodes (step 534). In a preferred environment, the operation of this step either returns a flag or some other value to indicate whether the timing information was correctly installed on the slave node. This flag or value is verified in step 536. If the video timing information was correctly installed, then the function or procedure on step 530 is complete. Otherwise, the system may be configured to determine whether a compatible video timing is available (step 538). If not, the system may be configured to remove that particular node from the graphics processing and rendering process of the graphics for that particular display. Otherwise, the compatible timing information or data may be utilized (step 539) and installed in the graphics node (step 534).

Once the graphics nodes devices have been configured, then, as illustrated in FIG. 19, the graphics node configuration files are configured (step 540). This step is illustrated in further detail in FIG. 22. In this regard, the graphics node configuration files are configured by allocating and retrieving slave options (step 542) transferring these options to the various slave computers (step 544). Then, for each slave computer, each specific configuration file is generated, based up on the retrieved slave options (step 546). This step is essentially the generation of the individual slave configuration files, as was discussed in connection with Figs. 16 and 17. Alternatively, the slave configuration could